ARL-TR-7434 ● SEP 2015

US Army Research Laboratory

A Generalized Method for Vertical Profiles of Mean Layer Values of Meteorological Variables

by James Cogan

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-7434 ● SEP 2015

US Army Research Laboratory

A Generalized Method for Vertical Profiles of Mean Layer Values of Meteorological Variables

by James Cogan
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
Sep 2015	Final	02/2012–08/2015

4. TITLE AND SUBTITLE
A Generalized Method for Vertical Profiles of Mean Layer Values of Meteorological Variables

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
James Cogan

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
US Army Research Laboratory
ATTN: RDRL-CIE
2800 Powder Mill Road
Adelphi, MD 20783-1138

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-7434

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
In the first decade of this century, numerical weather prediction (NWP) models were first used in an artillery meteorological (MET) system to produce MET messages. The development and testing of the new system required a means to generate MET messages from the NWP model output and produce the same types of messages from radiosonde-based data for test purposes. The Army has fielded 2 major upgrades since that first system, but the message generation methodology has remained much the same. The US Army Research Laboratory (ARL) developed and continues to update a generalized method to produce the several message types as well one or more user-defined "messages." In its current form, the software can ingest from 2 to as many as 4000 observations, and can produce a vertical "profile" of nearly 4000 integrated mean layer values from the surface to the highest level of the height structure of the "message." In this report, the primary methods are presented along with the major program routines in pseudo code and text, and samples of the input and output.

15. SUBJECT TERMS
MET messages, mean layer values

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			James Cogan
Unclassified	Unclassified	Unclassified	UU	62	19b. TELEPHONE NUMBER (Include area code) 301-394-2304

Contents

List of Figures

List of Tables

Acknowledgment

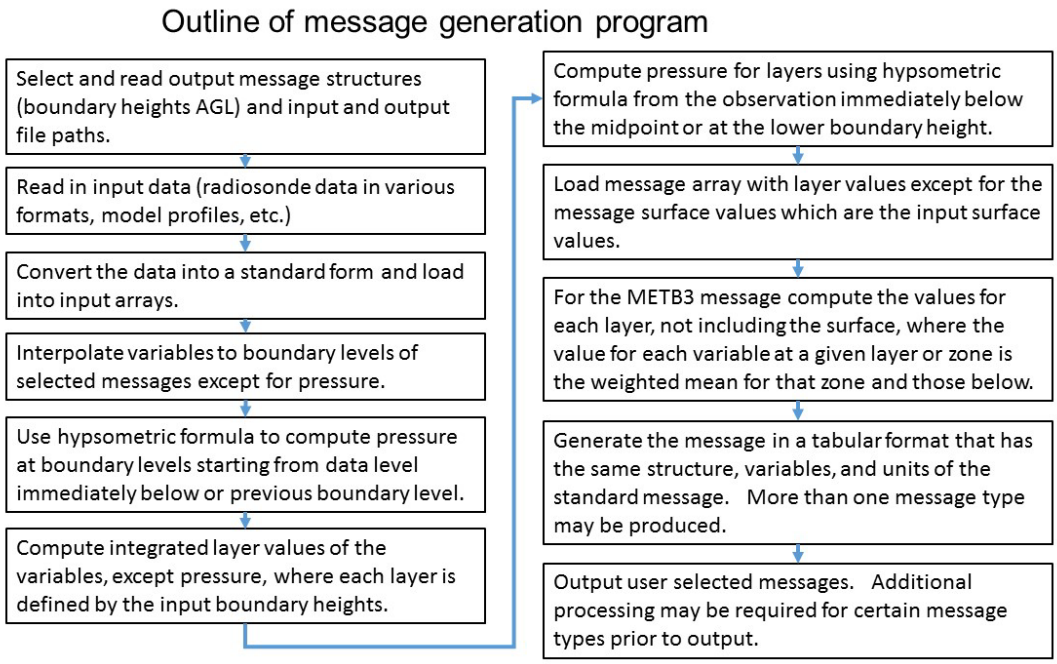
I would like to acknowledge Dr B Reen who provided the software to extract vertical profiles of meteorological variables of interest from Weather Research and Forecasting (WRF) output, as well as assistance on running WRF on local and high-performance computing (HPC) computers.

1. Introduction

In the first decade of this century, a numerical weather prediction (NWP) model, the Mesoscale Model Fifth Generation (MM5), was first used in an artillery meteorological (MET) system to produce MET messages. The development and testing of that first NWP-based system required a means to generate MET messages from the model output based "soundings" and produce the same types of messages from radiosonde-based data for test purposes. The Army has fielded 2 major upgrades since then. The first one essentially put the first system's software on a single laptop, but the more recent one hosts the current community model, the Weather Research and Forecasting (WRF) model, plus other improvements. However, the message generation methodology has remained much the same. The US Army Research Laboratory (ARL) developed a generalized method to produce several message types such as the commonly used Computer Meteorological Message (METCM) and the Ballistic Meteorological Message for surface to surface fires (METB3) as well one or more user-defined "messages." In its current form, the software can ingest from 2 to as many as 4000 observations, and can produce a vertical "profile" of nearly 4000 integrated mean layer values from the surface to highest data level that fits within the structure of the "message." In this report, the primary methods that form the overall procedure are presented along with the major program routines in pseudo code and text, and samples of the input and output.

2. Outline of Method

The original version of the method is described in Cogan and Jameson (2004). The current code is in the "C" language as in the original version. Some of the basic algorithms have either not changed or only have relatively minor modifications. Some of the main routines (also called functions in C) are repeated in the form of "pseudo code" in Appendix A of this report. An overall main function reads in the input file name from the command line and calls the various components as needed. Figure 1 presents an outline of the principal components of the software.



Note: AGL = above ground (or surface) level.

Fig. 1 Block diagram of message generation program showing main sections

The variables and format change from one type of input to another (e.g., radiosonde- vs. model-generated profiles) as expected, but also frequently change within the same type of system (e.g., different version of radiosondes from the same manufacturer). Also, different organizations often present their data differently even though the source may be the same. Given the wide variation in input, the program was designed so that the program essentially consists of 2 main sections, a smaller one to read the input data and convert them into a "standard" format, and the much larger one to process the data and produce the requested messages. The standard format in this case refers to a common set of variables and a single format for use by the processing part of the program. The first 3 boxes of Fig. 1 relate to this first part. The other boxes refer to the main part of the program where the large majority of the processing takes place. The types of messages that may be generated include the METCM, the target area low-level MET message (METTALL), METB3, and a user-defined message (USRMSG). In addition, considerable additional processing is needed for the METB3 during the preparation of the output. The following sections briefly describe the main sections of the message generation software. Appendix A contains abbreviated text and pseudo code descriptions of the major functions.

2.1 Setup Input and Parameter Files

Various parameter files initialize several aspects of the program. The input_parameters file tells the program where to find the input data file and where to place the output message file(s). The user may select the same or different directories for the input and output files.

A series of parameter files define the message structures. For example, the file metcm_lvls contains the boundaries of the METCM zones starting at the surface (zone 0) up through the highest zone (zone 31 with an upper boundary of 30000 m). The program determines which messages to produce depending on whether or not it finds the appropriate _lvls file. If none are found, the program ends and prints an error message. These files provide the structure information for several messages including, for example, the METCM (metcm_lvls), METTALL (tall_lvls), basic wind message (bwind_lvls), and the METB3 (balis_lvls). In addition, users may define their own "message" by listing the boundary levels in the file named "usrmsg_lvls". In order to save a structure file for later use, one can simply change the name and the program will ignore it (e.g., changing balis_lvls to balis_lvls1).

Other parameter files are used as needed. Because the METB3 message contains relative rather than absolute representations of the atmospheric parameters, additional tables with the standard profiles and weighting values are required during the computations. The METB3 message if selected requires parameter files in addition to balis_lvls. The file bal_std defines the standard atmosphere values for the computation of a METB3. The METB3 also needs weighting tables for density, temperature, and wind (bal_weights_dens3, bal_weights_temp3, and bal_weights_wind3). If no METB3 is computed, then the program ignores these files.

2.2 Read the Data and Convert to the Standard Format

The function to read the input data generally will change for the different types of input, but the rest of the program most often will not need any modification. Each new type of input data will require a recompilation of the code. However, since each of the several parts is in a separate source code file, only 1 file needs to be changed for a new type of input. For example, the file readwyo6.c contains the latest version of the readraob function for input of radiosonde data in the format of the soundings found on the University of Wyoming's weather website (http://www.weather.uwyo.edu), and the file readwrf9.c has the latest version of the readwrf function for input of vertical profiles of MET variables derived from WRF model output. In the program's current form, the input data such as a

radiosonde sounding must be an ASCII flat file (2-D table format). All versions of the input function read in the data and enter them into a structure of arrays with a "standard" format. Note that radiosonde-based soundings have a variety of formats and WRF-based "soundings" can have different formats as well. Nevertheless, once the sounding has been read in to the standard array, the rest of the program remains the same for either radiosonde- or WRF-based input.

2.3 Interpolate to the Boundary Levels of the Message Layers

The main program calls a function to compute the values for the output message layers. That function (msgvalues) first interpolates the variables to the message levels, which are the boundaries of the message layers. Those levels are defined in the appropriate lvls parameter file. This first part for the computation of the values at the boundaries of the message layers is described in this section. A generalized level function interpolates from the surface to the highest message level that is covered by the input data (except for pressure), that is, all levels for layers where the upper boundary level/height lies within the input data heights. For example, if the uppermost data levels or heights are 9367, 9785, 10007, and 10678 m, the highest METCM layer will be that from 9000 to 10000 m (zone 16). In this example, the last interpolation occurs for the level between 9785 and 10007 m (10000 m), and data at 10678 m are ignored. Table B-1 lists the boundary levels for the METCM. The interpolation function, level.c, is presented in Appendix A in the form of a descriptive pseudo code with text.

Pressure is computed using the standard hypsometric formulation (definition in Glickman [2000]) from the data level immediately below the relevant message level. If the 2 heights are the same, then the pressure value at the message level is the same. The exception to the procedure occurs for the surface where the level value is the input surface value. The mean virtual temperature for the computation of pressure is the average of the value at the aforementioned data and message levels. In rare cases, the gap between input data levels may exceed 2 or more layers (zones), although for all radiosonde soundings and model output profiles to date that has not occurred. In that case for a given level, one can use the interpolated virtual temperature and computed pressure for the message level immediately below in lieu of the observation. A few tests using actual radiosonde soundings where data between about 925 and 150 mb were removed showed that the differences in the resultant METCMs were on the order of 1 mb. No difference was found for smaller gaps (e.g., between 700 and 400 mb), and differences were found for only a couple of zones for an intermediate gap (between 925 and 400 mb).

2.4 Compute the Integrated Mean Values for the Message Layers

The second part of the aforementioned msgvalues function computes the layer or zone values using the input data and the values at the boundary levels. The procedure essentially provides an integrated or weighted mean of the layer or zone where the layer may have from 0 to n observations (data levels) and n may exceed 1000. The exception is the surface where the surface values are the values for zone 0. The generalized layer function, layer.c, computes the integrated mean layer or zone values for each of the several variables except for pressure. Figure 2 illustrates the procedure for a single layer that contains several observations. Appendix A contains a description of layer.c in the form of pseudo code with text as used for the presentation of level.c.

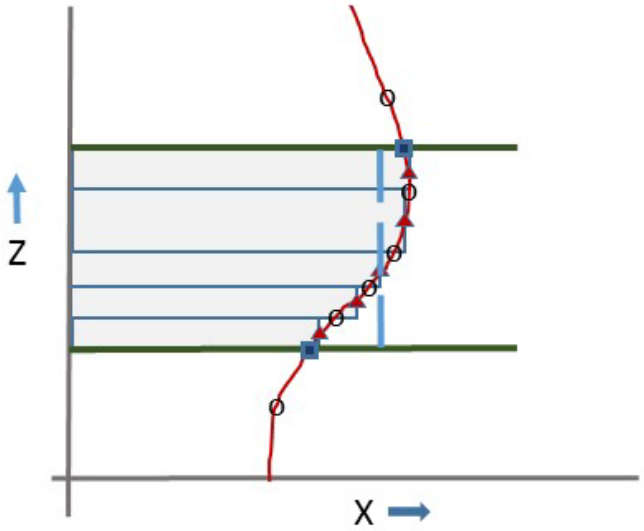


Fig. 2 Illustration of the computation of a weighted or integrated mean layer or zone value. The red curved line is a vertical (Z) profile of some variable (X). Circles, triangles, and squares represent, respectively, observations, mean values of sub-layers, and the interpolated values at the layer or zone boundaries. The vertical dashed line represents the layer or zone value.

As with the interpolation, the pressure values are computed separately with the hypsometric formula within the msgvalues parent function. In an earlier version of the software, the layer value of pressure was computed for the midpoint of the layer or zone using the virtual temperature and pressure values at the lower boundary of the zone and the zone virtual temperature value computed previously using the layer function. The current method uses the pressure and virtual temperature values at the data level immediately below the midpoint of the layer if one exists between the midpoint and lower boundary; if such a level does not exist, it reverts to the original method. Figure 3 illustrates the data level heights from where the pressure calculation starts for the current and original (fallback) method. The current method

still uses the zone virtual temperature value as in the original method. For radiosonde sounding observations (RAOBs) and model-generated soundings to date, many, but not all, layers contain at least 1 input data level between the lower boundary and midpoint. For other sources of input with widely separated data levels, or where message zones are close together, a large number of layers may not contain an input data level between the lower boundary and midpoint. An example of a message with thinner layers or zones is the METALL. USRMSGs may have very fine layers, perhaps as thin as a few meters.

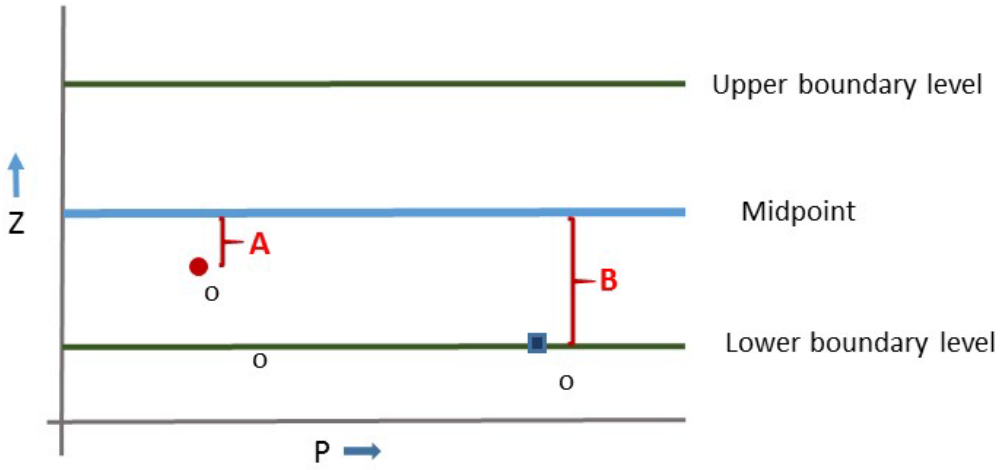


Fig. 3 Illustration of the data level heights where pressure calculations start for computation of layer values of pressure. The bracket denoted by "A" for the current method indicates that the pressure for the layer starts at the observation point immediately below the midpoint. That denoted by "B" for the original and now the fallback method illustrates the use of the pressure value at the lower boundary level/height of the layer.

The differences in computed pressure between the original method and the current one are minor, usually hundredths to tenths of a mb that normally would not appear in a METCM where pressure is to the nearest whole mb. However, even a small difference (e.g., < 0.01 mb) very near the n.5 mb point, where n is any whole number, may change the listed value by 1 mb (hPa). As expected, when no observation occurs between the lower boundary and midpoint of a zone, the computed pressure is the same from both methods. Also, radiosonde manufacturer's literature suggests overall accuracies on the order of ± 0.5 to 1 mb (hPa) suggesting that the differences are almost always within the measurement accuracy of the radiosonde pressure sensors. Consequently, the aforementioned differences tend to be less than the uncertainty in the measurements. The differences for a sample of METCMs only appeared for a few layers where the gap in data levels was very large. For example, pressure values for 2 of the 31 METCM zones (zones 13 and 14; midpoints at 6500 and 7500 m) differed by 1 mb for an input radiosonde sounding with data removed between 925 and 400 mb.

2.5 Output the Selected Message Types

The output functions produce tables that have the same variables, units, and structure as the selected message types. The format may not be exactly the same. For example, the METCM output has the same variables, units, and zones of a standard METCM, but not necessarily the exact same format. Since the General Trajectory (GTRAJ) program from the Armaments Research Development and Engineering Center (ARDEC) requires a specific format that is not the same as the one used in the field, the output routine produces METCMs that have the GTRAJ input format. Also, additional variables may be added as needed for evaluation purposes, etc. If additional minor processing is needed to, for example, change units for use with different systems, then that is performed within the output routine. The appropriate output routines are called from the main program when it finds the corresponding _lvls parameter files. Following the computation of the layer values, they are entered directly into the output array except for the METB3 message, which requires additional processing (see next section). Appendix C presents some samples of output from the program for a few of the message types including a USRMSG. FM 3-09.15/MCWP 3-16.5 (2007) also contains information on some of the standard types of messages.

2.6 Compute the METB3 Message Values

The METB3 output function (writebaliss) calls a separate METB3 function (bal_met), which uses the previously computed individual layer values as input. Other input is read from parameter files. One has the standard atmospheric values for wind speed, sensible temperature, and density. Three others have, respectively, 2-dimensional (2-D) arrays of weights for 1) wind speed, 2) sensible temperature, and 3) density. The standard atmosphere used is based on the International Civil Aviation Organization (ICAO) 1976 Standard Atmosphere, which is identical to the US Standard Atmosphere below 32 km (http://ccmc.gsfc.nasa.gov/modelweb/atmos/us_standard.html). The 2-D arrays of weights may be found in FM 6-16 (1979) and STANAG 4061 (2000). STANAG 4061 also has a table of additional values for temperature versus the use of constant values for the highest zones, i.e., zones 9–15.

The METB3 function first reads in the standard atmospheric values of wind speed, temperature, and density for the METB3 layers, followed by the 2-D arrays for the same variables (see Appendix B). The next process computes the layer values of density, and then converts temperature and density to the respective ballistic message units, that is, as a percentage of standard. The function uses the u and v wind components in knots, not wind speed and direction. The resultant layer values

of the variables are then weighted using the appropriate tables where the value of a variable at a given layer is the weighted value for the current layer and from all layers below excluding the surface values. The wind components are weighted with the values in the wind speed table. The exception to this method is the surface where the values are the surface values converted to ballistic message units. The wind speeds and directions are then computed from the components. If a radiosonde supplies the input the wind speed and direction have units of degrees and knots, which requires the corresponding values for the computation of wind components and conversion to speed and direction. Some wind input from other sources may have mils or tens of mils for direction (6400 mils in a circle or 1° ~ 17.778 mils), or m/s for speed. The bal_met function has slightly different versions for the different types of input that take into account those differences.

2.7 Alternative Method for Computation of METB3 from METCM

Cogan and Sauter (2013) describe a version of a method used at ARDEC as part of the software for a Marine handheld device. Ray (2013) at ARDEC provided a spreadsheet that contains their method. Both versions share the same basic algorithms. The main difference between this method and that used in the message-generation program of this report (Sections 2.1–2.6) is in the calculation of the individual layer values prior to weighting. In the Cogan and Sauter (2013) method, where the METB3 and METCM layers or zones are the same (e.g., zone 2 where both extend from 200 m to 500 m), the values are copied to the respective METB3 layers. Where they are not the same, the values are linearly interpolated from the midpoints of the 2 adjacent METCM layers above and below the METB3 layer midpoint. The procedure is illustrated in Cogan and Sauter (2013) and reproduced here as Fig. 4.

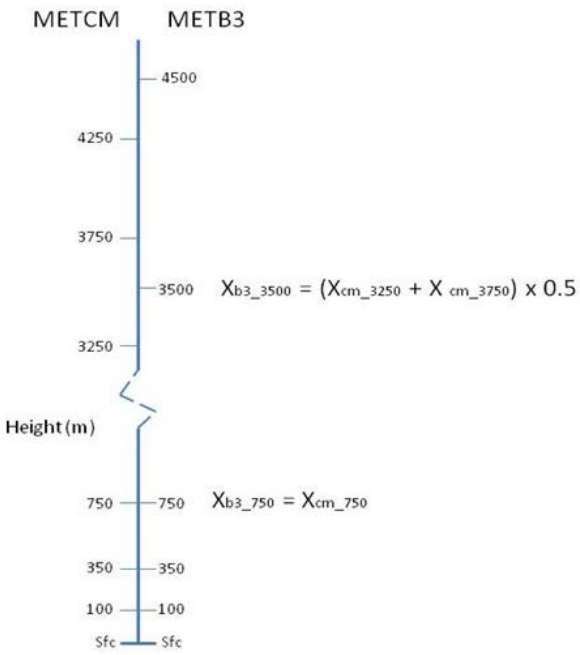


Fig. 4 Illustration of the METCM and METB3 zone midpoint heights and the method of obtaining METB3 zone values from METCM zone values (from Cogan and Sauter [2013]). "X" represents any variable and the subscripts (b3 for METB3 and cm for METCM) indicate the message type and midpoint. All heights are meters AGL.

The zone weighting method after computation of the individual layer values is the same. A separate program contains this alternate method and is not part of the main message generation software. Over a dozen comparisons between the ARL and ARDEC versions of the alternate conversion program produced the same results, with 1 exception. Some values of density were very slightly different at the highest zones. The reason was a slight difference in the standard value of density for zone 12 (11000 m midpoint) between the value in the ARDEC spreadsheet and the value in the ARL program. The ARL value more closely matches that in STANAG 4061 (2000), and is within a tenth of a g/m^3 of the STANAG value.

3. Input Soundings and Output MET Messages

The 2 most frequently used inputs as of the time of this report consist of RAOBs from the University of Wyoming Weather website (http://www.weather.uwyo. edu) and vertical profiles derived from WRF model output. The location specified in the MET message for RAOB input is the location of the launch site. For WRF input, the location used in the MET message is derived from the model grid in the WRF output. The WRF sounding consists of columns that have the values of the MET variables in the vertical for a selected model grid location or a location between

grid points where the column values are interpolated horizontally from the nearest grid points. In the separate program used to extract the column from WRF output developed by Reen (2015), the user can select the nearest grid point, a bilinear interpolation, or inverse distance weight interpolation. For non-complex terrain and benign weather conditions, the 2 interpolation methods yield nearly the same result, especially for fine-scale output (e.g., 1-km grid spacing). If the selected location is very near a grid point, then the results from the 2 interpolation methods are essentially the same and are close to output from the nearest grid point. Appendix C contains a sample of RAOB- and WRF-based input.

The output files contain the same structure, variables, and units of the respective MET messages, but may have a different layout and additional variables for comparison purposes. The current program generates more than 1 form for some of the message types. For example, 1 of the types of METCM output has sensible temperature as well as virtual temperature. The METCM output for input to GTRAJ has only the METCM variables. The USRMSG allows the user to design a message with a different structure and with the basic or derived atmospheric variables. Samples of the several message types are presented in Appendix C along with additional information.

4. Conclusion

This report describes a program developed at ARL to convert RAOB and NWP model (i.e., WRF) output based files into the several types of standard MET messages as well as user-defined output. The program has been used in various tests and analyses including developmental tests for artillery MET systems, development of MET Error Budget information as related to messages derived from NWP output, and analysis of WRF-based soundings compared to co-incident RAOBs. While the basic form of the program has not changed, specific functions have evolved over time to more efficiently produce the needed messages, generate additional forms of the messages, respond to customer requirements for modified versions, and deal with new input types or formats.

The program has other uses, as well. The USRMSG provides a means to compare output from different models or configurations of a single model at various altitudes and vertical resolutions. An example could be fine resolution near the altitude of the tropopause or within the boundary layer with a coarser resolution elsewhere. The output vertical layer resolution can be as fine as a fraction of a meter, though the user should consider the vertical resolution of the input when deciding on the number and thickness of individual layers. A very fine vertical spacing in the output may not yield much additional value if the data levels of the input are much further

apart. However, the program can use MET data from a variety of sources with only modification of the input function. Analyses of model output are planned that could have application for civilian as well as military users. Those analyses will be the subject of future reports.

5. References

Cogan J, Sauter D. Generation of ballistic meteorological messages - surface to surface (METB3s) from computer meteorological messages (METCMs). Adelphi (MD): Army Research Laboratory (US); 2013. Report No.: ARL-TN-0550.

Cogan J, Jameson T. Meteorological message and test analysis software for an army meteorological system. Adelphi (MD): Army Research Laboratory (US); 2004. Report No.: ARL-TR-3249.

FM 6-16, Department of the Army, Tables for Artillery Meteorology (Electronic) Ballistic Type 3 and Computer Messages, 1979, Washington, DC.

FM 3-09.15/MCWP 3-16.5, Department of the Army, Tactics, Techniques, and Procedures for Field Artillery Meteorology, 2007, Washington, DC.

Glickman T (Managing Ed). Glossary of Meteorology (2nd Ed), 2000, American Meteorological Society, Boston, MA.

Ray C. Private communication, 2013.

Reen B. Private communication, 2015.

STANAG 4061 (Edition 4), North Atlantic Treaty Organization - Military Agency for Standardization, Adoption of a Standard Ballistic Meteorological Message, 2000, Brussels, Belgium.

STANAG 4082 (Edition 2), North Atlantic Treaty Organization - Military Agency for Standardization, Adoption of a Standard Artillery Computer Meteorological Message, 2000, Brussels, Belgium.

Appendix A. Program Description

This Appendix provides the primary routines or functions (C code terminology) within the program for generation of meteorological (MET) messages or other layered structure specified by the user. The program was written in "C", but here the descriptions are in text and "pseudo code" form. The 2 key functions level and layer, respectively, interpolate to the message data levels and compute the integrated or weighted mean values of each variable for the message layers. They are the same for all versions of the program. Figure 2 in the main text illustrates the procedure for a single layer that has several observations. These are presented first followed by the other primary functions. Although there are different versions for the different types of input (e.g., radiosonde data from the University of Wyoming Weather website (http://www.weather.uwyo.edu) or vertical profiles derived from Weather Research and Forecasting (WRF) model output [Appendix C]); the several functions are essentially the same except for those that read the input data. Even the read data functions are very close except for the specific code that reads the different headers and the different formats of the input. The version that reads radiosonde data in a format used on the University of Wyoming web site is presented here (the format is referred to as "Text: List" on the website and is described in Appendix C).

The basic level and layer functions are the first ones to be described (Section A-1). They are followed by the other primary functions with the main function at the end, which organizes and ties the several functions together. An additional program was developed that translates a Computer Meteorological Message (METCM) into a Ballistic Meteorological Message for surface to surface fires (METB3) (Section A-2). Terms enclosed between "/* and "*/" are comments and a semicolon is used to end most lines as in "C". Some descriptive text ends with a period. The primary data structure used in the several functions contain data arrays plus site information. The data structure definition is presented in Appendix B (Table B-3).

A-1 Program for Generation of MET Messages

A. Level function:

This function is used in the generation of any pre-defined atmospheric structure of heights and layers. It ingests values at one set of height levels and produces values at another set of height levels. This description follows the one in Cogan and Jameson (2004). Note that the output data structure uses the terminology "message level[i] value" for i^{th} output level values in line with usage in the program. An index value of 0 represents the surface. This function assumes the height of the first input data level is the same as that of the first output data level (e.g., both are at the surface). The first output data level (zone 0) for all the message types is for the

surface. The goal is to use the input sounding values at the respective heights to create the level values at the user-defined levels.

Pass in the following variables: the maximum height/level of useful input data + 0.001m, the maximum size of the output arrays/user input levels, the array of user defined levels/upper boundary heights of layers plus the surface, the array of input sounding values, the array of output values (level values) for the defined levels, and the array of input sounding heights.

```
{
set indices to initial values of i=0, j=0;

    while(input sounding height[j] < maximum height and user defined level[i] < maximum height
and i < maximum size of output arrays)
        {
           if(user defined level[i] ≥ input sounding height[0])
              {                          /*Avoid negative heights or missing data (i.e., ERROR = -999).*/
                 if(user defined level[i] = sounding height[j])  /* No interpolation needed.*/
                    {
                       set level value[i] to sounding value[j];
                       add 1 to j;
                     }
                 else
                    {                                /* Linear interpolation with respect to height.*/
                       if(sounding height[j] > user defined level[i])
                          subtract 1 from j;
                       level value[i] = sounding value[j] – (sounding value[j] – sounding
value[j+1])* (user defined level[i] – sounding height[j])/(sounding height[j+1] – sounding
height[j]);
                     }
              }
            while (user defined level[i+1] ≥ sounding height[j+1] and sounding height[j+1] >
0.00001)
                 add 1 to j;
             add 1 to i;
         }
     return to calling program;
}
```

B. Layer function:

This function is used in the generation of any pre-defined atmospheric structure of heights and layers. It produces weighted mean (a.k.a. integrated mean) values for layers bounded by the user-defined levels. This description follows the one in Cogan and Jameson (2004). Note that the output data structure uses the terminology "message layer[i] value" for the i^{th} output layer value in line with the program coding. An index of 0 represents the surface. The temporary arrays are used only within the function.

```
Pass the following variables: the maximum height, the maximum size of input/temporary arrays
(size), the maximum size of the output arrays/user input levels, the array of user defined
levels/upper boundary heights except for the surface, the array of input sounding values, the array
of level values (level values) for the user defined levels as computed from the level function, the
output array of layer/zone values (layer values), and the array of input sounding heights.
{
Declare indices i=0, j, ind.
Define temporary value, height, and mean value arrays.
Define the variable sum.

while (i < maximum size of output arrays -1 and user defined level[i+1] ≤ maximum height)
   {                                                                          /* Find layer means.*/
       add 1 to i;
       set ind = 0 and sum = 0;
       temporary value[ind] = level value[i-1];                  /* Lower boundary value*/
       temporary height[ind] = user defined level[i-1];        /* Lower boundary level*/

       for (j starting at 0 increasing to a value not to exceed 'size' incremented by 1)
          {
             if (sounding height[j] < user defined level[i] and
                          sounding height[j] > user defined level[i-1])        /* Values within a layer.*/
               {
                   add 1 to ind;
                   temporary value[ind] = sounding value[j];
                   temporary height[ind] = sounding height[j];
                }
          }
        add 1 to ind;
        temporary value[ind] = level value[i];                   /* Upper boundary value.*/
        temporary height[ind] = user defined level[i];        /* Upper boundary level.*/

        for (j starting at 1 increasing to a value of ind and incrementing by 1)          /* Sub-layer
average.*/
           {
               mean value[j-1] = (temporary value[j] + temporary value[j-1]) * 0.5;
           }
        for (j starting at 1 increasing to a value of ind and incrementing by 1)
            {                                                    /* Proportional weighting of each layer.*/
               add mean value[j-1] * (temporary height[j] – temporary height[j-1]) to sum;
             }

         layer value[i-1] = sum /(user defined level[i] – user defined level[i-1]);   /* mean layer
value.*/

     }      /* End of while loop.*/
   return to calling program;
}
```

C. Msgvalues function:

This function is the largest and performs the computations of the values at the layer boundary levels and computes the integrated mean values of those layers. It incorporates and organizes the sequence of the level and layer functions, the pressure calculations, and the calculation of various parameters used within this

function and elsewhere in the program. This description includes text and pseudo code. In this description the terms for an array element use "[x]" as, for example, height[i] or within a data structure level[i] height. The parameter ERROR (= –999) indicates missing or bad data. Note that the resultant data structure (output from this function) uses the terminology "message level[i] value" for the i^{th} value.

Pass in the input data structure and the output data structure for the output layer values and the one for the computed level values.
/* Note that the structures are structures of arrays that hold header information as well as the data. Each level or layer is an element of an array that contains the values for that level or layer as structure elements. */
{
/* message values = layer values, but may have different indices, mlevel = level values */
Set the indices and the variable definitions.

number message heights = number level heights - 1; /* Number of layer values is one less than level values. */
size = number of input sounding heights;
msize = number of level heights;
zmax = maximum allowed height = sounding input data level height at index 'size -1' + 0.001;

Set the temporary structures used within this function.
/* For example, snd = (struct temporary *)malloc(sizeof(struct temporary));
and similarly for leveltemp and layertemp. */

Initialize structures and temporary variables with the missing data/bad data indicator. /* Missing or bad data indicated by the defined value ERROR = -999.0. */

Set parameters for level and layer values.

/* Compute components from input sounding wind speed and direction. */

start j at -1 /* Here j is used to count the number of levels with wind data.*/
for i from 0 to < size incrementing by 1 /*Here i is the index for the input sounding data.*/
{
if(input wind speed at i^{th} height not = ERROR and input wind direction at i^{th} height not = ERROR)
{
add 1 to j;
direction = -(input wind direction at height [i]) * PI/180 + 3* PI/2;
snd u-component at height[i] = cos(direction) * input wind speed at height[i];
snd v-component at height[i] = sin(direction) * input wind speed at height[i];

snd height[i] = input sounding height at input data level[i];
}
}
wsize = j; /*wsize is the number of input levels with wind data. */

/* Compute temperature and virtual temperature. Temperature converted from C to K as needed in calling main function. */

/* Compute level values. */

for i from 0 to < size incrementing by 1

```
    {   /* tvfromtemp computes virtual temperature (Tv) from pressure (P), sensible temperature
(T), and relative humidity (H) using a standard method. */
       virtual temperature at height[i] of input = tvfromtemp(temperature, pressure , relative
humidity at height[i]);
    }

/* Set up temporary variables for use in level and layer functions as needed. */
  start j at -1    /* Here j is used to count the number of levels with T, P, and H data.*/
  for i from 0 to < size incrementing by 1    /*Here i is the index for the input sounding data.*/
    {
     if(height, T, and P at input data level[i] not = ERROR)
       {
         add 1 to j;
         snd Tv[i] = input sounding Tv at input data level[i];
         snd T[i] = input sounding T at input data level[i];
         snd P[i] = input sounding P at input data level[i];
         snd H[i] = input sounding H at input data level[i];
        /* Height values for snd computed earlier.*/
       }
     }
  tsize = j; /*tsize is the number of input levels with T,P, and H) data. */

  for i from 0 to < msize incrementing by 1 /* Initialize temporary level and layer heights.*/
    {
      leveltemp height[i] = upper boundary level[i] height;
      layertemp height[i] = leveltemp height[i];
    }

  /*Compute level values for T, Tv, and H where the level function is described in this appendix.*/

  level(zmax, msize, leveltemp height array, snd T array, leveltemp T array, snd height array);
/*In the function description the respective variables or arrays are zmax, htsize, zh, value, lev-
value, z. */
  level(zmax, msize, leveltemp height array, snd Tv array, leveltemp Tv array, snd height array);
  level(zmax, msize, leveltemp height array, snd H array, leveltemp H array, snd height array);

 /*Compute layer values for T, Tv, and H where the layer function is described in this appendix.*/

  layer(zmax, tsize, msize, layertemp height array, snd T array, leveltemp T array, layertemp T
array, snd height array);
/*In the function description the respective variables or arrays are zmax, size, htsize, zh, value,
lev_value, lay_value, z. */
  layer(zmax, tsize, msize, layertemp height array, snd Tv array, leveltemp Tv array, layertemp Tv
array, snd height array);
  layer(zmax, tsize, msize, layertemp height array, snd H array, leveltemp H array, layertemp H
array, snd height array);

  /* Compute Wind Speed and Direction */

  /* Compute level values of wind components (u, v). */

  level(zmax, msize, leveltemp height array, snd u array, leveltemp u array, snd height array);
  level(zmax, msize, leveltemp height array, snd v array , leveltemp v array, snd height array);

  /* Compute level wind speed & direction from components (u, v). */
```

```
  for i from 0 to msize incrementing by 1
    {
     level[i] value of wind direction = (2* PI - atan2(leveltemp u[i], -leveltemp v[i])*180/ PI;
     if(level[i] direction > 360)
       subtract 360 from level[i] direction;

     level[i] value of wind speed = sqrt(leveltemp u[i] * leveltemp u[i] + leveltemp v[i] *
leveltemp v[i]);
    }

  /* Compute layer values of components (u, v). ********************************/

   layer(zmax, wsize, msize, layertemp height array, snd u array, leveltemp u array, layertemp u
array, snd height array);
   layer(zmax, wsize, msize, layertemp height array, snd v array, leveltemp v array, layertemp v
array, snd height array);

  /* Compute message layer wind speed & direction from components (u, v). Message values are
the message layer or zone values in this function. */

    surface message wind speed = input surface wind speed;   /* i =0 at surface*/
    surface message wind direction = input surface wind direction;
    surface message u = input surface u;
    surface message v = input surface v;

   for i from 1 to < msize incrementing by 1
     {                              /* Here message level[i] value is the resultant layer[i] value. */
       message level[i] u = layertemp u[i-1];
       message level[i] v = layertemp v[i-1];

       message level [i] wind direction = (2* PI - atan2(message level[i]u, -message level[i]
v))*180/PI;
       if (message level[i]direction > 360)
         subtract 360 from message level[i] direction;

       message level [i] wind speed = sqrt(message level[i] u * message level[i] u + message
level[i] v * message level[i] v);
      }

 /* Load height values into the message structures. */

  for i from 0 to < msize incrementing by 1
    {
      message level[i] height = level[i] height;   /* 'message level' refers to resultant layer, 'level'
refers to upper boundary level of layer */
    }

 /* Load T, Tv, H values into level and message structures. */

  for i = 0 to < msize incrementing by 1          /* level values */
    {
      level[i] T = leveltemp T[i];
      level[i] Tv = leveltemp Tv[i];
      level[i] H = leveltemp H[i];
    }
```

```
  message T at surface = input surface T;        /* Message values for surface, where i =0. */
  message Tv at surface = Tv from surface data;
  message H at surface = input surface H;

  for i from 1 to i < msize incrementing by 1
   {
     message level[i] T = layertemp T[i-1];    /* Message values above surface. */
     message level[i] Tv = layertemp Tv[i-1];
     message level[i] H = layertemp H[i-1];
   }

  /* Pressure values using better hypsometric method. Takes into account possible situation of
input with few observation levels and large vertical gap(s) between observation levels. Not seen to
date in either radiosonde or model output based soundings. */
/* Compute layer pressure values. */

  level pressure for surface = input surface pressure value;

  set j = 0;

  for i from 1 to < msize incrementing by 1                {
    while(snd height[j] < level[i] height)
     {
       add 1 to j;
     }

    if(snd height[j] == level[i] height)
     level[i] P = snd P[j];
    else
     {

       if(sound height[j-1] < level[i-1] height and j > 0)
         {
           level[i] P = presscomp(level[i] Tv, level[i-1] Tv, level[i-1] P, level[i] height, level[i-1]
height);
         }
       else
         {
           level[i] P = presscomp(level[i] Tv, snd Tv[j-1], snd P[j-1], level[i] height,
snd height[j-1]);
         }

     }

   }

  /* Compute layer pressure values.
*****************************************************/

  message P at surface = level P at surface;
  set j=1;

  for i from 1 to <msize incrementing by 1
   {
    while(snd height[j] < average of level[i] height and level[i-1] height)
       add 1 to j;
```

```
      if(snd height[j] = average of level[i] height and level[i-1] height)
        {
         message level[i]P = snd P[j];
        }
      else
        {
         if(snd height[j-1] < average of level[i-1] height and level[i-2] height)
           {
            tv = message level[i] Tv;
            tv0 = level[i-1] Tv;
            z = average of level[i] height and level[i-1] height;
            z0 = level[i-1] height;

         /* presscomp is the function for computation of pressure using a standard method. */
            message level[i] P = presscomp(tv, tv0, level[i-1] P, z, z0);
           }
         else
           {
            tv = message level[i] Tv;
            tv0 = snd Tv[j];
            z = average of level[i] height and level[i-1] height;
            z0 = snd height[j-1];

            message level[i] P = presscomp(tv, tv0, snd P[j-1], z, z0);
           }
        }
    }

   /* Eliminate incorrect values at top of message where missing data occurs at the top of the input
sounding using the msg_mod function (not described here). Values eliminated in groups: P, T, and
H, and wind speed and direction. Normally not called since the read function usually handles this
issue. */

     if(wsize != tsize)
       msg_mod(input sounding array, message array);

   /* Load in site information (date, time, lat, lon, etc.). */

   message site information = input site information;

 /*Free up temporary arrays in order to release memory. */

    free(snd);
    free(leveltemp);
    free(layertemp);

 return to calling function;
}
```

D. Readraob function:

This function reads in data in the format used at the University of Wyoming. The other read functions are very similar and are not presented here. The parameter ERROR (=–999) indicates missing or bad data. This description includes text and pseudo code. Here raob refers to radiosonde observation (RAOB) input.

```
Declare the several variables. /* Includes temporary variables such as the array temp.*/

Pass in the input data structure and the input data file name.
/* The file name includes the path. Data are entered into the input data structure in this function.*/
{
Open the input data file.

/* Read the input data. */
/* Read header information. */

Set header ceiling and visibility to ERROR. /* Radiosonde data normally do not include these
                                                parameters.*/
Read in header information. /* Includes name of site, time, day, latitude, longitude, and elevation.
*/

/* Read data lines. */

/* Check for missing data at beginning. Can occur when site above height of standard levels. In
that case have extrapolated height (MSL) and pressure (e.g., 150 m and 1000 mb when the site
elevation is 350 m. */

for j from 0 to < 3 incrementing by 1
   read from input file jth temp value;   /*temp refers to a temporary array.*/
increment i by 1;

while(temp  [2] > 299.99)  /* Check for value  ≥ 300 units. It will find the lowest data level that
contains temperature data in degrees C which will be < 300 C vs. pressure data that should be ≥
300 hPa or mb. (see Table C.1 for a sample of relevant input).*/
    {
        increment i by 1;
        set temp [0] to temp [2];
        for j from 1 to < 3 incrementing by 1
            read from input file j^th temp value;
    }

set raob surface pressure to temp [0];
set raob surface height to temp [1];
set raob surface temperature at temp [2];
read from input file the remainder of the first data line. /*index i = 0*/

set index i = 1;

while(when read a line of raob variables the number of input data items for each line = 11) /*Data
items are pressure, height, temperature, dewpoint, humidity (relative), wind direction, wind speed,
plus four not used for MET message.*/
   {
     if(raob wind speed at level [i] > 221)
       {
```

```
            print message saying found apparent excessive wind speed;
            print wind speed for level [i];
            subtract 1 from i;
        }
       add 1 to i;
     }

if(raob height [i] < raob height [i-1] and  raob height [i] not = ERROR)
    {
        print the index i, height [i] and height [i-1];
        set the number of raob lines to i-2;
        print the message that the number of lines was changed based on > 1 missing line at end of
sounding;
    }
else
    set number of raob lines to i;

/* Convert MSL heights to AGL as needed. **********/

      if(the raob surface height > 0)                /* Always true so far for U of Wyoming listings. */
        {                                                    /* But just in case ... */
          raob site elevation  = raob height at surface;
          data set is labeled as MSL;
         }
      else
        data set is labeled as AGL;

     if(data set is MSL))     /* Change MSL to AGL. Not used for AGL. */
       {
        if(raob surface height < raob site elevation) /* Check for raob surface vs. elevation values. */
          raob site elevation = raob surface height; /* Should not be necessary, but could happen. */

        if(raob surface height > raob site elevation)
          height difference = raob surface height - raob site elevation;
        else
         height difference = 0;

        for i from 0 to < number raob heights incrementing by 1 /*This loop for MSL heights only.*/
           raob height [i] = raob height [i] – (raob site elevation + height difference);
       }

/* Get latitude, longitude, and elevation from information section that follows data columns. */

find latitude and read it in;
find longitude and read it in;
find elevation and read it in;

close the input file;

return to calling function;
}
```

E. Writemetcm function:

This function prints the METCM data in up to 3 forms each in a separate file. All the forms have the structure (i.e., 32 zones from 0 through 31) and units of the METCM, but may not have the same numerical format. The main output has header information such as day/time and location, and includes additional variables (i.e., sensible temperature). The second file has the data in a format suitable for input to the Armaments Research Development and Engineering Center's (ARDEC) General Trajectory (GTRAJ) model. This version of the message discards the header information as well as the additional variables. A third "test" version has the variables and form of the second (GTRAJ) version. It is included to allow the user to change the precision of the output (e.g., pressure from whole units to hundredths) for use when testing software changes, etc. In normal use, the appropriate section of code for the test version can be commented out or deleted. This description includes text and pseudo code.

With the exception of the writebaliss function described below, the write functions for the other message types (not shown) have the same structure and design even though, for example, column labels and the specific data format of the output variables may be different. In some cases not all variables are output (e.g., basic wind report).

Pass in the output data structure and the path for the output. /* Note that the structures are structures of arrays that hold header information as well as the data. */
{
Declare the variables and output file names.

Generate the complete output file names. /*file name and path*/

Open the output files.

/*Set the parameter for the output loops.*/
size = number of METCM heights + 1;

/* Print the main file with METCM values. **********/

Print the header information. /* Date, time, latitude, longitude, elevation, ceiling, and visibility:
If ceiling and/or visibility are not available then the "no data"
indicator is used (-999). */

Print the column headers for line or zone number, height (upper zone boundary), wind direction, wind speed, virtual temperature, pressure, and sensible temperature.

for i from 0 to < size incrementing by 1
{
if wind speed [i] not equal to ERROR /* ERROR is the missing data indicator (= -999).*/
{
wind speed [i] = wind speed [i] * 1.9438; /*knots per m/s; if in knots comment out or delete */

```
                wind direction [i] = wind direction [i] *1.77778; /* Units of mils*10 per degree; if
already in mils*10 comment out or delete. */
            }

        if virtual temperature [i] not equal ERROR
            virtual temperature [i] = virtual temperature [i] * 10; /* METCM units of tenths of a
degree expressed as a whole number. */

        if temperature [i] not equal ERROR  /* temperature is the sensible temperature vs. virtual*/
            temperature [i] = temperature [i] * 10;  /* METCM units of tenths of a degree (e.g.,
291.5 K becomes 2915 K*10). */

        print to file values of i, wind direction [i], wind speed [i], virtual temperature [i], pressure,
and temperature [i];    /* here i is the same as the zone number */
   }

 /* Print the METCM file in GTRAJ input format. ***********/

 for i from 0 to < size incrementing by 1
   {
      set the zone number = index i;
      print to file values of zone, wind direction [i] , wind speed [i], virtual temperature [i],
pressure [i];
   }

 /* Print the test version with variables of GTRAJ input; output formats of variables can be varied.
*/

 for i from 0 to < size incrementing by 1
   {
      set the zone number = i;
      print to file values of zone, wind direction [i] , wind speed [i], virtual temperature [i],
pressure [i];
   }

 close output files;

 return to calling function;

}
```

The functions for radiosonde and model output based input soundings essentially are the same. Code for computing the test version of the METCM may or may not be included depending on the need. The functions for the other messages are very similar as well with the exception of the one for the METB3. The METB3 write function follows.

F. Writebaliss function:

This function includes additional code not found in any of the other write functions. It contains the call to the bal_met function for computation of the METB3 values from the individual layer values and changes the units from the usual MET units to

the relative units used in the METB3 message. For example, temperature changes from degrees to percent of the standard atmosphere value. This description includes text and pseudo code. ERROR (= –999) is the missing or bad data indicator. Also, balis refers to arrays for individual layer values and balmsg refers to arrays for METB3 type values.

```
Pass in the output data structure and the path for the output.
{
    Declare the variables and output file names.

    set the balmsg temporary structure;

    generate the complete output file name;    /*file name and path*/

    open the output file;

    call the bal_met function;                    /* computes METB3 weighted values */

    for i from 0 to < number of ballistic message heights + 1 incrementing by 1
       {
          if(u component [i] or pressure [i] equals ERROR) /*Removes bogus line at top of
message if it occurs. */
             set height [i] = ERROR;


      /* Convert to METB3 units and print METB3 values. ***********/

    print the header information; /* Date, time, latitude, longitude, elevation */

    print the column headers for height (upper zone boundary), line or zone number, wind
direction, wind speed, temperature, and density.
/* The units of temperature and density are percent of standard in tenths of a percent expressed as
a whole number (e.g., 98.7 % becomes 987). Also for values ≥ 100 % the value is decremented by
100% and then expressed as a whole number of tenths of a percent or in other words the leading 1
is dropped with leading zeros retained (e.g., 102.8% becomes 028 and 100.3 % becomes 003). The
units of wind direction is in hundreds of mils and wind speed is in knots. */

    while(i < number balis heights +1 and balmsg wind direction [i] is not < 0  and balis height [i]
not = ERROR)
       {
          if(balmsg wind speed [i] is not < 0)   /*Needed in case of calm (wind speed=0)
conditions.*/
             {
                balmsg windspeed [i] =  balmsg windspeed [i] * 1.9438; /* Only use if input in m/s.
*/
                balmsg wind direction [i] =  balmsg wind direction [i] * 1.7778; /* Only use if
input in degrees. */
                balmsg wind direction [i] =  balmsg wind direction [i] * 0.1; /* Convert to hundreds
of mils. */
             }

          if(balmsg temperature [i] > 999.500)
             subtract 1000 from balmsg temperature [i];
```

```
            if(balmsg temperature [i] < 0.499)
                balmsg temperature [i] = 0;

            if(balmsg density [i] > 999.500)
                subtract 1000 from balmsg density [i];
            if(balmsg density [i] < 0.499)
                balmsg density [i] = 0;

          print to file balis height [i], i (= zone number), balmsg wind direction [i], balmsg wind
speed [i], balmsg temperature [i], and balmsg density [i];

          increment i by 1;

      }

   free the temporary balmsg structure;
   close the output file;

   return to the calling function;

}
```

G. Bal_met (ballistic MET) function:

This function is used in the generation of the METB3. It produces weighted layer or zone values using the individual layer values computed earlier in the program. This listing is an abbreviated version, that is, some lines are left out or summarized. In this description, some actual code is presented as well as pseudo code or text.

Pass in the values for the individual layers for the ballistic message that are input for the output METB3 message. Pass out the weighted layer or zone values of the METB3.
{
Define the file names and file properties for the files holding the standard atmosphere values.
Define the arrays for the tables of wind, temperature, and density, plus for the separate variables.
/*The definition for a 2-D array in C is complex if “released” before the program ends so that memory is preserved. The maximum layer or zone number is tabnum = 15 starting at 0 (16 zones). The actual code for the density table (balwt_d) is shown; the others are the same except for the variable names.*/

```
float **balwt_d

  balwt_d = (float **)malloc(tabnum * sizeof(float *));  /* Table for density*/
  if(balwt_d == NULL)
    {
     printf("balwt_density2: INSUFFICIENT MEMORY!\n");
     exit(1);
    }
  for(i=0;i<tabnum;i++)
     {
       balwt_d[i] = (float *)malloc(tabnum*sizeof(float));
       if(balwt_d[i] == NULL)
         {
           printf("balwt_density: INSUFFICIENT MEMORY!\n");
           exit(1);
```

```
        }
    }
```

Open the standard ballistic MET files. /* (1) the file with standard atmosphere values and (2-4) the weighting tables for wind, temperature, and density.*/

```
  }
```

Set default (no data = ERROR = -999) values for the ballistic output message array (balmsg) and the ballistic weight arrays (tables).

Read the standard ballistic met values (temperature and density) /*Note: standard value for wind speed is 0. */

Compute individual layer values in ballistic MET units and compare with the standard atmosphere values to find percent of standard.

Compute wind components. /* Wind values normally in knots and degrees or mils (or tens of mils). */

/* Compute weighted layer values for type 3 ballistic messages (surface to surface). */
/* balwt_d, balwt_t, and balwt_w refer to the weights for density, temperature, and wind. */

```
  for i from 1 to i < tabnum+1 incrementing by 1
   {
    Set sums for each variable at zero.  /*For example, sumd = 0.*/
    for j from 0 to j < i incrementing by 1
     {
      add layer density value [j+1] * density weight [i-1][[j] to sumd;
      add layer temperature value [j+1]* temperature weight [i-1][j]) to sumt;
      add layer u-component value [j+1]* wind speed weight [i-1][j]) to sumu;
      add layer v-component value [j+1]* wind speed weight [i-1][j]) to sumu;
     }
   /*Load message values with sums */
    balmsg density [i] = sumd;
    balmsg temperature [i] = sumt;
    balmsg u-component [i] = sumu;
    balmsg v-component [i] = sumv;
   }

 /*  Load surface values. Surface value has index 0.*/
  balmsg density [0] =  density [0];
  balmsg temperature [0]  = temperature [0];
  balmsg u-component [0] = u-component [0];
  balmsg v-component [0] = v-component [0];
```

/* Compute ballistic wind speeds and directions from weighted components. */

```
  balmsg wind speed [0] = input wind speed [0];
  balmsg wind direction [0] = input wind direction [0];
```

for i from 1 to i<tabnum+1 incrementing by 1 /*Compute speed and direction (using atan2 function). */

```
   {
     balmsg wind direction [i]= (2* PI - atan2(u-component [i], v-component [i]))*180/ PI;
```

```
      if (balmsg wind direction [i] > 360)    /* This version used for radiosonde input (knots and
degrees). */
         subtract 360 from balmsg wind direction;

      balmsg wind speed [i] = sqrt(balmsg u-component [i] * balmsg u-component [i] + balmsg v-
component [i] * balmsg v-component [i]);
    }
```

Close the files containing the standard values and the weight tables for density, temperature, and wind.

```
/* Free or release arrays: allows re-use of memory. */

/*Free the 2-D arrays.*/
for i from 0 to i < tabnum incrementing by 1)
   {
    free(balwt_d[i]);
    free(balwt_t[i]);
    free(balwt_w[i]);
   }
 free(balwt_d);
 free(balwt_t);
 free(balwt_w);

 free the 1-D arrays; /* e.g., free(lpd) */

 return to calling function;
}
```

H. Main function:

This function (filename begins with convertdata) organizes and calls the main routines of the program that read, process, and output data, opens most of the input and output files, sets up the primary arrays, initializes those arrays, calls a function for gross error checks, sets up the requested message height structures, and reads in the name of the input file from the command line. The versions for radiosonde and model output based soundings are essentially the same with only very minor differences. Here we describe the function for radiosonde input. This description is in text and pseudo code. ERROR (=–999) is the missing or bad data indicator.

```
Obtain the name of the input file from the command line.
{
Declare the variables and structures for the level and layer values for the several message types.
/* The structures contain the level and layer arrays where each element of an array contains the
several variables as defined in the relevant include file. The structures are declared in such a way
that their memory allocation can be released. */

Initialize the various structures.

Open the parameter file (input_parameters).
If no file present print an error message and exit program.

Open message height files and set respective message height file indicator if file present.
```

/* metcm_lvls, balis_lvls, bwind_lvls, etc., and cmswitch, balisswitch, bwindswitch, etc. These files define the structure of the respective messages (surface and upper boundary heights). The "switches" tell the program whether or not to call the functions for the selected messages.*/

If no height or lvls files are found then print the message "No met messages computed. Need height structure input files" and exit the program.

Read the input and output file paths from the input_parameters file.
Append the input file name from the command line to the input path to get the complete name.

Read the input data via the readraob function. /* Described above in this section.*/

Perform a gross error check.

/* Set up the message heights. **********/

/* METCM */

if(cmswitch = 0) /* If file available for METCM structure (i.e., metcm_lvls); yes if = 0. */
{
set i to 0;
while(read (metcmlevel level[i] height) = 1 and metcmlevel level[i] height ≤ snd level[last -1] height)
/* The read (fscanf in C) = 1 if it finds 1 value. The index "last" refers to the last or highest input level. The data structure for the values at the output data levels (boundaries of zones except the surface or zone 0) is metcmlevel and snd is the input data structure. */
{
add 1 to i;
}
number of metcmlevel data levels = i;
print a message with the number of METCM data levels;
}
/* Follow the same procedure for the other message types – not repeated here. *****/

/* Computation section */

Convert input temperatures from C to K; /* Not needed if input already in K. */

/* Compute values for the selected types of meteorological messages. */

/* METCM if selected uses the msgvalues function described above in this section to produce values for the METCM layer data structure. */
if(cmswitch = 0)
call msgvalues (input data structure, METCM layer data structure, METCM level data structure);

/* Additional calls to msgvalues for the other message types follow the same type of syntax (e.g., msgvalues(input data structure, METALL layer data structure, METALL level data structure): Not repeated here. */

/* Generate output. */

/* METCM if selected uses the appropriate writemetcm function as described above in this section. */

```
        if(cmswitch = 0)
          {
             call writemetcm (METCM layer data structure,  output file path);
             close the METCM height file;
          }
     /* The same type of calls used for the other message types that were selected – not repeated
here. */

/* Free memory of the several files. */

    free(snd);
    free(metcm);
    free(metcmlevel);
/* and similarly for the other output and boundary level files.*/

Close the parameter file that contains the input and output file paths.

} /* End of the program. */
```

A-2 Program for Conversion of METCM to METB3

ARL also provided software for conversion of a METCM to a METB3 (Cogan and Sauter 2013). The difference between computing a METB3 from a METCM and deriving it directly from radiosonde or model data is in the computation of the individual layer values. In the case of converting from a METCM, the midpoint values of the METCM layers are used vs. integrated mean values. As before, the description includes text and pseudo code. ERROR (=–999) is the missing or bad data indicator.

```
Obtain the input file name.
{

Declare the variables and data structures. /*Input METCM, ballistic message layer, and ballistic
message level structures.*/

Initialize the structures.

Open the parameter file.
  if cannot open
    {
       print error message;
       exit program;
     }

read the input path;
append the file name;

Open input data file.
  if cannot open
    {
       print error message;
       exit program;
     }
```

```
Open METB3 zone heights file. /*Surface height (0 m) and heights of upper boundary of zones.*/
  if cannot open
   {
      print error message;
      exit program;
    }

Call the readmetcminput routine to read the input METCM. /* Similar to readraob described in
section A.1.*/

Check for out of bounds data. /* Gross error check. If out of bounds sets to ERROR. ***/

Check for missing or bad data input data. /* Checks for values = ERROR: procedure seldom
invoked. */
If missing/bad data found print message to alert the user. /* Will cause incorrect output at heights
≥ that input level. */

Read in the METB3 zone heights. /* Surface (= 0) plus upper boundary heights. */

Find the number of METB3 zones covered by the METCM input.

Call the function to compute the individual layer values. /*Alternate version of msgvalues
function.  See paragraph in text below. */

Compute and write the message values. /* This function (writebalis) is the same as in A-1*/

Free the data structures. /*frees up memory*/

Close the parameter and input data files.

}
```

The function for computing the individual layer or zone values primarily differs from the one described in Section A-1 (msgvalues) in that the individual layer values are not the integrated or weighted mean values, but rather taken as the values at the METCM midpoints. Where the METCM and METB3 zones are the same (e.g., zone 1), then the METB3 value equals the corresponding METCM value. Otherwise, the value for the METB3 layer is linearly interpolated between the METCM zone with a midpoint immediately above the METB3 midpoint and that for the METCM midpoint immediately below. A mean or average is sufficient since the height difference is the same between a relevant METB3 midpoint and the METCM midpoints immediately above and below. No METCM zone is entirely within the bounds of a METB3 zone. Wind speeds and directions from the METCM zones are converted into their respective horizontal components (u, v) before being converted into METB3 zone values.

In this version of the function, pressure is computed in the same manner as the other variables, that is, equal to the METCM value or linear interpolation where the

respective METB3 and METCM zones do not match. Figure 4 in the main text illustrates the procedure.

A version of this program not described here allows the user to extrapolate to the maximum METCM zone when the input METCM has fewer than the number of zones for a complete METM3. The full extrapolated METCM is then used to produce the METB3. For the extrapolated zones the wind speed and direction are held constant from the last zone computed with actual wind data. To obtain the extrapolated temperatures and pressures the respective percentages of standard at the uppermost zone derived from actual data are multiplied by the respective standard atmosphere values for the extrapolated zones. For example, if P_T is the percentage of standard temperature at the last data derived zone, the temperature at zone X above is P_T times the standard value for zone X.

The function for reading the METCM, readmetcminput, is similar to the other input routines. Two versions exist at this time. One reads in METCMs in the "readable" text format produced by current artillery MET systems and the other reads in METCMs in the GTRAJ input format. For the first version, only certain header information is retained for later use in the output message. The function discards the remainder and other non-data information. The second version only reads the relevant data since the input file does not have a header or other non-data characters or numbers. The heights assigned to the data levels are the midpoints of the METCM and are read in from the appropriate lvls file (metcmin_lvls).

INTENTIONALLY LEFT BLANK.

Appendix B. Message Zones, Primary Data Structure, and Sample METB3 Weighting Array

This section presents various tables concerning message zones, the main data structure, and a sample Ballistic Meteorological Message for surface to surface fires (METB3) weighting array. Tables B-1 and B-2 show the zone structures for the Computer Meteorological (MET) Message (METCM) and METB3, respectively. The others (not shown) are similar to the METCM in that a variable value for each zone above the surface represents the mean for that zone. Details on the MET messages may be found in the relevant Field Manual (e.g., FM 6-16) and STANAG (e.g., STANAG 4082). The METB3 differs in that the values for each zone above the surface are the weighted means of the specified zone and all lower zones except for the surface. For all types the surface or zone 0 values are for the surface only.

Table B-1 The boundary heights (levels) of the METCM. Except for the surface (zone 0), the height listed alongside the zone number is the upper boundary. For example, the upper boundary of zone 7 is 3000 m. The zone midpoint height for that zone is 2750 m. All heights are above ground level (AGL).

Zone	Height (m)
0	0
1	200
2	500
3	1000
4	1500
5	2000
6	2500
7	3000
8	3500
9	4000
10	4500
11	5000
12	6000
13	7000
14	8000
15	9000
16	10000
17	11000
18	12000
19	13000
20	14000
21	15000
22	16000
23	17000
24	18000
25	19000
26	20000
27	22000
28	24000
29	26000
30	28000
31	30000

Table B-2 The boundary heights (levels) of the METB3. Except for the surface (zone 0) the height listed alongside the zone number is the upper boundary. For example, the upper boundary of zone 6 is 3000 m. Unlike the other message types the METB3 zone values, except for the surface, are weighted mean values of the variable for the specified zone and all zones at lower levels above the surface. For example, the density value for zone 7 is the weighted value for zones 7 down through 1. All heights are AGL.

Zone	Height (m)
0	0
1	200
2	500
3	1000
4	1500
5	2000
6	3000
7	4000
8	5000
9	6000
10	8000
11	10000
12	12000
13	14000
14	16000
15	18000

The procedure for computing the values for a METB3 from a METCM is presented in Cogan and Sauter (2013) and Ray (2013) and the main routine is described previously in Section A-2.

The programs described in this report use a specified data structure (Table B-3) to hold input, output, and many intermediate forms of the data. The original version was developed at the former Environmental Technology Laboratory (ETL) of the National Oceanic and Atmospheric Administration (NOAA) and later modified at US Army Research Laboratory (ARL). This data structure is presented here in its "C" code format. It is defined within the metstruct5.h include file, which, in turn, is accessed via the main convert5.h include file. MAXSIZE currently is defined to be 4000. This data structure also is used in other related programs not described here (e.g., for extrapolation of a model derived profile down to a lower surface level), and consequently includes some variables or parameters not needed for the program described in this report.

Table B-3 The primary data structure (sound) used in the programs of this report

```
/* sounding (normally WRF or raob output) structure */
struct sound
{
 int nht;                          /* number of heights or data levels */
 struct site_t site;               /* info on the site */
 struct level_t level[MAXSIZE]; /* data for each height (up to MAXSIZE heights) */
};

/* info on the site */
struct site_t
{
 char filename[81]; /* name of file from PPDF or other source */
 char date[51]; /* year month day */
 char time[21]; /* hour min sec */
 char time_id[4]; /* time zone (e.g., GMT or UT) */
 char hgt_id[4];  /* MSL or AGL */
 float lat; /* latitude */
 float lon; /* longitude */
 float elev; /* elevation */
 float prs; /* pressure */
 float ceil; /* ceiling or cloud base */
 float vis; /* visibility */
 float snowrate;  /* not used in extrapolation method, but retained here */
 float rainrate;      /* not used in extrapolation method, but retained here */
 float refract_index;   /* not used in extrapolation method, but retained here */
 float extrap_depth;  /* extrapolation height or depth of extrapolation layer */
 float max_extrap;  /* maximum height for extrapolation calculations */
 float minlapse;  /* minimum lapse rate for detection of "inversion" (may be > 0 by small amount)
*/
 float grid_sp;  /* grid spacing of input data */
 float hmax;  /* maximum height (AGL) for influence of surface wind (extrapolation) */
 float hmin;  /* minimum height (AGL) for wind computations */
};

/* data values for each height for upper level soundings */
struct level_t
{
 float hgt; /* height */
 float prs; /* pressure */
 float tmp; /* temperature */
 float vtmp; /* virtual temperature */
 float hum; /* humidity */
 float dew; /* dewpoint */
 float spd; /* wind speed */
 float dir; /* wind direction */
 float u;   /*u-component */
 float v;   /*v-component */
 float dens; /*density */
 int lvlnum; /* level from 01 to n */
 char htype[4]; /* AGL or MSL (currently use AGL) */
};
```

The METB3 variables for zones above the surface are weighted values of the variables at the given zone and all zones below, except the surface (zone 0). Table B-4 presents the array for density; the arrays for wind speed and temperature have a similar form and are used in the same way. For example, the density for zone 3 uses individual layer values in percent of standard from layers 1–3. The METB3 value of density for zone 3 therefore is the sum of 0.22 * d[1], 0.31 * d[2], and 0.47 * d[3] where the numbers in brackets represent the individual layer densities (d) computed using the msgvalues function described earlier.

Table B-4 The density weight array for computation of the METB3 message. The METB3 density (in percent of standard) for a given zone, with zones read from top to bottom starting at zone 1, is the sum of the densities for individual layers times the respective weights read from left to right (see text preceding this table for an example).

1.00													
.43	.57												
.22	.31	.47											
.15	.21	.32	.32										
.11	.17	.25	.22	.25									
.08	.11	.17	.17	.15	.32								
.06	.08	.14	.13	.12	.22	.25							
.05	.06	.11	.11	.10	.19	.17	.21						
.04	.06	.09	.09	.08	.17	.15	.14	.18					
.03	.04	.07	.07	.07	.13	.12	.11	.11	.25				
.01	.03	.05	.05	.06	.12	.11	.09	.09	.16	.23			
.02	.03	.05	.05	.05	.11	.10	.09	.08	.14	.12	.16		
.02	.02	.04	.05	.05	.11	.09	.09	.08	.14	.10	.09	.12	
.02	.03	.05	.05	.05	.10	.09	.08	.07	.13	.11	.08	.06	.08
.02	.04	.05	.05	.05	.10	.09	.08	.07	.12	.09	.08	.05	.05	.06

INTENTIONALLY LEFT BLANK.

Appendix C. Samples of Input and Output

C-1 Samples of Input

The most common input for the message generation program come from radiosonde soundings or observations (RAOBs) or are derived from the Weather Research and Forecast (WRF) output. The radiosonde- and WRF-based soundings often come in different formats. Here we present the 2 most common ones at this time. The University of Wyoming provides RAOBs in a readily understood format (e.g., Table C-1) and the WRF version shown here (Table C-2) was developed by Reen (2015). The WRF sounding may be extracted via a script using the National Center for Atmospheric Research Command Language (NCL; http://dx.doi.org/10.5065/D6WD3XH5) from the vertical profiles at the nearest grid point or use 1 of 2 methods for horizontal interpolation to the selected location, inverse distance weighting or bilinear. With the finer grid resolutions and under fairly benign conditions over non-complex terrain, the differences between the interpolation methods are very small, and the output values are close to those derived from the nearest grid point. Where the chosen location is very near the grid point the differences in the values from the three methods will be very small for most situations.

Table C-1 Example of RAOB for Albuquerque, New Mexico. The first 3 mandatory levels are below the actual terrain surface and consequently have only the pressure level and the extrapolated mean sea level (MSL) heights.

72365 ABQ Albuquerque Observations at 00Z 02 Jun 2015

PRES hPa	HGHT m	TEMP C	DWPT C	RELH %	MIXR g/kg	DRCT deg	SKNT knot	THTA K	THTE K	THTV K
1000.0	44									
925.0	746									
850.0	1495									
838.0	1619	30.2	3.2	18	5.78	250	7	319.1	338.0	320.2
831.0	1694	28.0	2.0	19	5.35	247	7	317.5	334.9	318.5
818.1	1829	26.7	1.6	20	5.27	240	8	317.5	334.7	318.6
789.8	2134	23.7	0.6	22	5.10	260	10	317.6	334.3	318.6
				.						
				.						
				.						
5.5	35442	-28.5	-69.5	1	0.60	114	11	1081.8	1090.9	1082.2
5.3	35662	-28.2	-69.2	1	0.64	130	10	1092.8	1102.7	1093.2
5.1	35966	-27.7	-68.7	1	0.71	45	18	1108.1	1119.2	1108.6
5.0	36126	-27.5	-68.5	1	0.76			1116.2	1128.0	1116.7

Station information and sounding indices

Station identifier: ABQ
Station number: 72365
Observation time: 150602/0000
Station latitude: 35.04
Station longitude: -106.62
Station elevation: 1619.0
Lifted index: -1.01
.
.
.
Mean mixed layer mixing ratio: 5.28
1000 hPa to 500 hPa thickness: 5836.00
Precipitable water [mm] for entire sounding: 16.96

Note: The variables used in the program of this report are pressure (PRES), height (HGHT), temperature (TEMP), relative humidity (RELH), wind direction (DRCT), and wind speed (SKNT). The program extracts the station ID, name, and date/time from the header line and station latitude, longitude, and elevation from the list following the data columns. Dewpoint (DWPT) is stored in the input data structure, but is not used at this time. The variables MIXR (mixing ratio), THTA, THTE, and THTV are not used.

Table C-2 Example of a "sounding" derived from WRF output for Albany, New York

```
 42.6945  -73.8239     78.98 2015-06-15_06:00:00   1000.0   42.6900  -73.8300     0
   78.98 1004.98   18.37   94.99    5.38  160.22
   91.33 1003.46   18.45   93.41    4.92  160.20
  124.35  999.62   18.64   92.29    7.75  162.30
  174.12  993.85   18.56   92.32   10.12  165.02
  232.51  987.11   18.34   92.63   12.03  167.22
  299.68  979.42   18.00   93.86   13.28  168.93
  375.84  970.76   17.97   97.79   14.62  174.98
  457.16  961.62   18.49  100.00   14.25  180.97
  552.57  951.01   18.82  100.00   13.21  185.69
                     .
                     .
                     .

19934.39   58.07  -59.78    3.74    7.64   11.11
20248.98   55.22  -59.16    3.19    6.24   26.68
20522.86   52.85  -59.29    3.05    5.26   39.26
20752.06   50.95  -58.92    2.77    4.97   49.77
```

Note: The program extracts the output site latitude (deg), longitude (deg), elevation (m), date, time, grid spacing (m), requested latitude (deg), requested longitude (deg), and type of horizontal interpolation from the header line. The interpolation choices are 0 for the nearest grid point with no interpolation, 1 for inverse distance weighting, and 2 for bilinear. For the 2 latter choices the output and requested latitudes and longitudes will be the same. The data columns are height (MSL); pressure; temperature (C); relative humidity (%); wind speed (knots); and wind direction (deg).

C-2 Samples of Output

The Computer Meteorological (MET) Message (METCM) is the most commonly used message at this time. Here 2 versions of a METCM are presented, one with header information and column labels (Table C-3) and the other with only the data columns in a specific format for use as input to the General Trajectory (GTRAJ) model (Table C-4). The former one with header also has an additional column for sensible temperature. The GTRAJ version not only has fewer columns, but the order is different as well. The zone structures for both follow that of the standard METCM as listed in Table B-1 and seen in Table C-3. FM 3-09.15/MCWP 3-16.5(2007) also contains detailed information on several of the message types.

Table C-3 Example of a METCM with header and column labels produced from the RAOB shown in Table C-1. The header contains values derived from the Albuquerque, New Mexico, sounding for the date shown.

METCM output

Date: 02Jun2015 Time: 00Z Latitude: 35.04000 Longitude: -106.62000
Elevation: 1619.0 Ceiling: -999.0 Visibility: -999.0

Line	Height (m)	Wind Direction (tens of mils)	Wind Speed (kt)	Virt Temp (K*10)	Pressure (mb)	Temperature (K*10)
0	0	444	7	3045	838	3034
1	200	436	7	3022	829	3012
2	500	445	9	2995	805	2985
3	1000	448	10	2955	769	2946
4	1500	419	15	2906	725	2898
5	2000	426	15	2857	684	2850
6	2500	402	13	2810	644	2803
7	3000	403	9	2764	606	2757
8	3500	400	4	2717	569	2710
9	4000	484	4	2671	533	2665
10	4500	588	5	2636	501	2631
11	5000	640	4	2596	469	2593
12	6000	532	13	2570	425	2569
13	7000	480	25	2497	371	2497
14	8000	479	17	2410	323	2410
15	9000	493	17	2326	280	2326
16	10000	471	27	2241	241	2241
17	11000	493	19	2158	206	2158
18	12000	510	24	2114	176	2114
19	13000	508	23	2080	149	2080
20	14000	534	25	2059	127	2059
21	15000	560	22	2062	107	2062
22	16000	571	18	2097	91	2097
23	17000	581	12	2093	77	2093
24	18000	78	4	2114	66	2114
25	19000	41	10	2122	56	2122
26	20000	110	9	2137	48	2137
27	22000	122	8	2156	37	2156
28	24000	125	13	2192	27	2192
29	26000	176	10	2218	20	2218
30	28000	111	7	2280	15	2280
31	30000	102	5	2317	11	2317

Note: Virt Temp refers to virtual temperature. Since no ceiling or visibility information appeared in the input, the values shown (–999) represent missing data.

Table C-4 Example of a METCM with only the standard data columns. This version is ready for input to the GTRAJ model and has no header or column labels. The data columns are zone number, wind direction, wind speed, virtual temperature, and pressure. The values and units match those of Table C-3.

0	444	7	3045	838
1	436	7	3022	829
2	445	9	2995	805
3	448	10	2955	769
4	419	15	2906	725
5	426	15	2857	684
6	402	13	2810	644
7	403	9	2764	606
8	400	4	2717	569
9	484	4	2671	533
10	588	5	2636	501
11	640	4	2596	469
12	532	13	2570	425
13	480	25	2497	371
14	479	17	2410	323
15	493	17	2326	280
16	471	27	2241	241
17	493	19	2158	206
18	510	24	2114	176
19	508	23	2080	149
20	534	25	2059	127
21	560	22	2062	107
22	571	18	2097	91
23	581	12	2093	77
24	78	4	2114	66
25	41	10	2122	56
26	110	9	2137	48
27	122	8	2156	37
28	125	13	2192	27
29	176	10	2218	20
30	111	7	2280	15
31	102	5	2317	11

The Ballistic Meteorological Message for surface to surface fires (METB3) has a different format from the METCM and some listed variables are different. For example, temperature has units of percentage of standard where the standard is the 1976 International Civil Aviation Organization (ICAO) standard atmosphere. The message does not contain pressure values, but does have the derived variable density, also in units of percentage of standard. As with the METCM the METB3 has wind direction, but in units of hundreds of mils vs. tens of mils. Perhaps more significant, for each variable a given zone value in the METB3 is the weighted mean of the value for that zone and the values for all lower zones except the surface (zone 0). The weighting tables used to generate the METB3 may be found in FM 6-16 (1979) and STANAG 4061(2000). The METB3s computed by the message generation program contain header information and data of a standard METB3 in

the appropriate zone structure and units. Table C-5 is a sample METB3 computed directly from the Albuquerque, New Mexico, RAOB of Table C-1.

Table C-5 METB3 computed from the RAOB of Table C-1

```
METB3 output

Input filename: ABQ_2015060200

Date: 02Jun2015
Time: 00Z
Latitude:  35.04000
Longitude: -106.62000
Elevation: 1619.00
```

Height (m)	Line	Wind Direction (100s of mils)	Wind Speed (kts)	Temperature (pcnt std)	Density (pcnt std)
0	0	44	07	056	783
200	1	44	07	051	787
500	2	44	08	048	789
1000	3	45	09	045	792
1500	4	43	13	041	795
2000	5	43	13	038	798
3000	6	41	12	031	804
4000	7	42	07	026	809
5000	8	46	05	021	813
6000	9	49	08	019	815
8000	10	48	15	019	820
10000	11	48	16	019	827
12000	12	49	16	019	828
14000	13	50	17	019	832
16000	14	51	16	019	831
18000	15	51	14	019	829

The program also produces other message types including a user-defined "message". The other types are the Target Area Low Level MET (METTALL), Target Acquisition MET (METTA), and the Basic Wind Report (BWR). The user-defined message was given the name USRMSG. The METTALL from this program has additional variables: height, virtual temperature, and pressure. Tables C-6 through C-9 provide samples of those messages. The messages formatted as GTRAJ input are not shown.

Table C-6 METTALL message derived from the sounding of Table C-2

TALL output

Date: 2015-06-15
Time: 06:00:00
Latitude: 42.69450
Longitude: -73.82390
Elevation: 78.98

Line	Height (m)	Virt Temp (K*10)	Pressure (mb)	Wind Speed (kts)	Wind Direction (10's of mils)	Temperature (K*10)	Rel Humidity (pcnt)
0	0	2937	1005	10.5	285	2915	95
1	50	2939	1002	12.0	287	2917	93
2	100	2940	996	17.8	292	2918	92
3	200	2937	988	22.9	297	2915	93
4	300	2934	976	26.8	305	2912	95
5	400	2939	965	27.9	318	2915	99
6	500	2944	954	26.2	328	2919	100
7	600	2944	943	24.8	335	2918	100
8	700	2939	932	25.0	340	2914	99
9	800	2933	921	25.7	344	2909	98
10	900	2927	910	26.2	345	2904	97
11	1000	2921	900	26.2	347	2898	97
12	1100	2915	889	26.1	348	2893	97
13	1200	2909	879	25.9	351	2887	97
14	1300	2903	869	25.7	353	2882	98
15	1400	2897	858	25.4	354	2877	98
16	1500	2892	848	24.9	356	2872	98
17	1600	2887	838	24.3	358	2867	98
18	1800	2880	824	23.3	361	2861	97
19	2000	2870	804	22.0	366	2851	98
20	2500	2853	771	20.7	368	2836	98
21	3000	2828	726	20.8	371	2813	94
22	3500	2796	683	21.7	396	2784	91
23	4000	2765	643	24.6	428	2754	95
24	4500	2745	604	32.3	448	2735	98
25	5000	2722	567	37.2	453	2713	100

Note: Temp refers to temperature (sensible), Virt Temp to virtual temperature, Press to pressure, and Rel Humidity to relative humidity.

Table C-7 METTA message derived from the sounding of Table C-2

METTA output

Date: 2015-06-15
Time: 06:00:00
Latitude: 42.69450
Longitude: -73.82390
Elevation: 78.98

Line	Height (m)	Temperature (K*10)	Humidity (pcnt)	Virtual Temp (K*10)	Pressure (mb)	Wind Speed (kts)	Wind Direction (tens of mils)
0	0	2915	95	2937	1005	10	285
1	50	2917	93	2939	1002	12	287
2	100	2918	92	2940	996	18	292
3	200	2915	93	2937	988	23	297
4	300	2912	95	2934	976	27	305
5	400	2915	99	2939	965	28	318
6	500	2919	100	2944	954	26	328
7	600	2918	100	2944	943	25	335
8	700	2914	99	2939	932	25	340
9	800	2909	98	2933	921	26	344
10	900	2904	97	2927	910	26	345
11	1000	2898	97	2921	900	26	347
12	1100	2893	97	2915	889	26	348
13	1200	2887	97	2909	879	26	351
14	1300	2882	98	2903	869	26	353
15	1400	2877	98	2897	858	25	354
16	1500	2872	98	2892	848	25	356
17	1600	2867	98	2887	838	24	358
18	1700	2863	97	2882	828	24	360
19	1800	2858	97	2877	819	23	362
20	1900	2854	97	2872	809	22	365
21	2000	2849	98	2867	799	22	367
22	2100	2844	98	2862	790	21	369
23	2200	2840	98	2857	781	21	369
24	2300	2836	98	2853	771	21	369
25	2400	2831	98	2848	762	20	368
26	2500	2827	97	2844	753	20	367
27	2600	2823	96	2839	744	20	367

Note: Temp refers to temperature and Humidity to relative humidity.

Table C-8 BWR message derived from the sounding of Table C-1 to show the maximum height of the BWR. Height was added to the BWR of this program, but normally is not part of the standard BWR.

BWR output

Date: 02Jun2015
Time: 00Z
Latitude: 35.04000
Longitude: -106.62000
Elevation: 1619.00

Line	Height (m)	Wind Speed (kts)	Wind Direction (tens of mils)
0	0	7	444
1	2000	12	431
2	4000	7	411
3	6000	8	551
4	8000	21	480
5	10000	22	479
6	12000	22	503
7	14000	24	522
8	16000	20	565
9	18000	7	611
10	20000	9	75
11	22000	8	122
12	24000	13	125
13	26000	10	176
14	28000	7	111
15	30000	5	102

Table C-9 USRMSG derived from the sounding of Table C-2. The user determines the structure, header information, units, and labels. Since no ceiling or visibility information appeared in the input, the values shown (–999) represent missing data. Future versions of the input files (WRF derived output) may contain this information, other parameters could replace the current ones, or the user may delete them without replacements. Consequently, any two USRMSG files may have different data types, structure, or format. Virt Temp refers to virtual temperature.

USER DEFINED output

Date: 2015-06-15 Time: 06:00:00 Latitude: 42.69450 Longitude: -73.82390
Elevation: 78.98 Ceiling: -999.0 Visibility: -999.0

Line	Height (m)	Wind Direction (degrees)	Wind Speed (kn)	Virt Temp (K*10)	Pressure (mb)	Temperature (K*10)
0	0	160	5	2937	1005	2915
1	50	161	6	2939	1002	2917
2	100	164	9	2940	996	2918
3	150	166	11	2938	990	2916
4	200	168	12	2936	985	2914
5	300	172	14	2934	976	2912
6	400	179	14	2939	965	2915
7	500	184	13	2944	954	2919
8	600	188	13	2944	943	2918
9	700	191	13	2939	932	2914
10	800	193	13	2933	921	2909
11	900	194	13	2927	910	2904
12	1000	195	13	2921	900	2898
13	1100	196	13	2915	889	2893
14	1200	197	13	2909	879	2887
15	1300	198	13	2903	869	2882
16	1400	199	13	2897	858	2877
17	1500	200	13	2892	848	2872
18	1600	201	13	2887	838	2867
19	1700	202	12	2882	828	2863
20	1800	204	12	2877	819	2858
21	1900	205	11	2872	809	2854
22	2000	206	11	2867	799	2849
23	2100	208	11	2862	790	2844
24	2200	208	11	2857	781	2840
25	2300	207	11	2853	771	2836
26	2400	207	10	2848	762	2831
27	2500	207	10	2844	753	2827
28	2600	206	11	2839	744	2823
29	2700	206	11	2834	735	2819
30	2800	207	11	2829	726	2814
31	2900	210	11	2823	717	2808
32	3000	213	11	2817	709	2803
33	3200	217	11	2807	696	2794
34	3400	224	11	2793	679	2780
35	3600	232	12	2779	663	2768
36	3800	239	12	2767	647	2756
37	4000	245	14	2757	631	2747
38	4500	252	17	2745	604	2735
39	5000	255	19	2722	567	2713
40	5500	256	20	2696	533	2687
41	6000	259	18	2666	500	2659
42	6500	269	15	2634	469	2629
43	7000	278	17	2599	439	2594
44	7500	279	18	2563	411	2560
45	8000	278	19	2529	384	2526
46	8500	276	19	2494	359	2492
47	9000	273	19	2457	335	2455
48	9500	271	19	2418	312	2417
49	10000	271	20	2377	291	2377

List of Symbols, Abbreviations, and Acronyms

2-D	2-dimensional
AGL	above ground level
ARDEC	Armaments Research Development and Engineering Center
ARL	US Army Research Laboratory
BWR	Basic Wind Report
ETL	Environmental Technology Laboratory
GTRAJ	General Trajectory
HPC	high-performance computing
ICAO	International Civil Aviation Organization
MET	meteorological
METB3	Ballistic Meteorological Message for surface to surface fires
METCM	Computer Meteorological Message
METTA	Target Acquisition MET message
METTALL	Target area low-level MET message
MM5	Mesoscale Model Fifth Generation
MSL	mean sea level
NOAA	National Oceanic and Atmospheric Administration
NWP	numerical weather prediction
RAOB	radiosonde sounding observation
USRMSG	user-defined message
WRF	Weather Research and Forecasting

1 DEFENSE TECHNICAL
(PDF) INFORMATION CTR
DTIC OCA

2 DIRECTOR
(PDF) US ARMY RSRCH LAB
RDRL CIO LL
RDRL IMAL HRA RECORDS MGMT

1 DIRECTOR
(PDF) US ARMY RSRCH LAB
RDRL CIE
J L COGAN

INTENTIONALLY LEFT BLANK.